Clase 1 — 15.10.25

🧠 Programación de Servicios y Procesos

Profesor: Jose Antonio Martín

UNIDAD: Programación Multiproceso

🗓️ Clase 1 — 15/10/2025

🎯 Tema: Tipos de archivos ejecutables y conceptos base

1️⃣ Objetivos

- **Mejorar el rendimiento y la eficiencia:** la programación multiproceso permite dividir tareas en procesos independientes que se ejecutan en paralelo, aprovechando múltiples núcleos de CPU.
- **Aumentar la escalabilidad:** al ejecutar múltiples procesos en paralelo, el sistema puede ampliar su capacidad sin perder estabilidad ni rendimiento.
- **Mejorar la capacidad de respuesta:** los sistemas multiproceso pueden atender múltiples solicitudes o tareas simultáneamente, lo que reduce la latencia y mejora la experiencia del usuario.
- **Aislamiento y seguridad:** cada proceso se ejecuta en su propio espacio de memoria, lo que evita interferencias entre programas y mejora la protección ante errores o vulnerabilidades.

> 💬 **Comentario del profesor:**
> Destaca la importancia del **número de núcleos del procesador**. Cuantos más núcleos tenga la CPU, mayor será la capacidad del sistema para ejecutar procesos en paralelo. La **programación multiproceso** busca precisamente aprovechar todos esos núcleos disponibles para maximizar la eficiencia.

2️⃣ Escalabilidad

Concepto:
La **escalabilidad** es la capacidad de un sistema para **mantener o mejorar su rendimiento** cuando se le añaden recursos. En el contexto de la programación multiproceso, implica que el sistema puede aprovechar más núcleos o hardware adicional para realizar más tareas de manera simultánea sin degradar su rendimiento.

Tipos de escalabilidad:

- **Vertical:** se mejora una máquina existente (añadiendo núcleos, RAM o velocidad de reloj).
- **Horizontal:** se añaden más equipos o procesos que colaboran entre sí (por ejemplo, servidores distribuidos).

Ejemplo práctico:
Un programa que está diseñado para dividir su trabajo entre 4 hilos podrá escalar bien en una CPU de 4 núcleos. Si la CPU tiene 8 núcleos, puede duplicar su rendimiento potencial siempre que el software lo soporte.

> 💬 **Comentario del profesor:**
> La escalabilidad **no depende solo del hardware**, sino también del **diseño del software**. Un programa

> que no ha sido preparado para ejecutar tareas en paralelo **no aprovechará los núcleos adicionales** del procesador.

3️⃣ Definición de Archivo Ejecutable

Concepto:
Un **archivo ejecutable** es un fichero que contiene **código compilado** listo para ser ejecutado por el sistema operativo. Este código instruye al sistema sobre cómo realizar una o varias operaciones.
El ejecutable puede incluir no solo el código máquina, sino también **recursos, bibliotecas y datos necesarios** para la ejecución del programa.

Ejemplos de extensiones ejecutables:

- **Windows:** `.exe`, `.bat`, `.dll`
- **Linux:** archivos sin extensión específica, marcados con permiso de ejecución (`chmod +x`)
- **macOS:** `.app`
- **Android:** `.apk`
- **iOS:** `.ipa`

Importante:
Llamar a todos los ejecutables ".exe" es incorrecto, ya que **.exe es un formato específico de Windows** (PE: *Portable Executable*).
Otros sistemas usan diferentes formatos como **ELF (Linux)** o **Mach-O (macOS)**.

> 💬 **Comentario del profesor:**
> Un **ejecutable es un programa o fichero que ejecuta el código de una aplicación**.
> Aunque a menudo se les llama ".exe", eso **solo aplica en Windows**.
> Además, los ejecutables **sirven para poner en marcha programas**, no para leerlos o editarlos directamente.

4️⃣ ¿Puede el usuario modificar un ejecutable?

Respuesta corta: No.
Los archivos ejecutables **no pueden modificarse directamente** porque están **compilados**. El proceso de compilación convierte el código fuente (legible por humanos) en código máquina (legible por la CPU), reorganizando estructuras internas, direcciones y dependencias.

Motivos:

- La información interna del ejecutable (rutas, direcciones, estructuras binarias) depende del compilador.
- Alterar bytes del ejecutable puede romper esas referencias, impidiendo que el sistema lo cargue correctamente.
- Muchos ejecutables están **encriptados u ofuscados** para proteger su integridad y evitar ingeniería inversa.
- Sin acceso al **código fuente original**, cualquier modificación directa es ineficaz o destructiva.

Ejemplo:
Editar un `.exe` con un editor hexadecimal puede alterar una dirección de memoria o una instrucción, provocando errores al ejecutarlo o bloqueando el programa.

> 💬 **Comentario del profesor:**
> Los ejecutables **no pueden ser modificados por el usuario** porque están **compilados por el autor del software**.
> Si se pudieran editar, se **rompería el programa**, ya que perdería sus rutas internas y dependencias.
> Además, muchos vienen **firmados digitalmente o cifrados**, y sin el **código fuente** no es posible modificarlos de forma útil o legal.

5️⃣ Aplicaciones Portables

Definición:
Una **aplicación portable** es aquella que **no requiere instalación** en el sistema operativo.
Puede ejecutarse directamente desde una unidad externa (por ejemplo, un pendrive o disco USB) sin escribir en el registro ni requerir permisos administrativos.

Características:

- Incluyen todos los archivos necesarios en la misma carpeta.
- Se ejecutan directamente sin instalador.
- No dejan rastro en el sistema (ideal para entornos de prueba o uso temporal).
- Al retirar la memoria externa, el programa ya no estará disponible en el equipo.

> 💬 **Comentario del profesor:**
> Una **aplicación portable** es aquella que **no necesita instalador**: simplemente ejecuto el fichero directamente desde el medio externo.
> Si retiro el USB o apago el PC, el programa desaparece del entorno.
> No debe confundirse con tener el código fuente, ya que sigue siendo un ejecutable compilado.

6️⃣ Tipos de Ejecutables por Sistema Operativo

Sistema Operativo	Extensiones / Formatos	Descripción
Windows	`.exe`, `.dll`, `.com`, `.bat`, `.pif`, `.cmd`, `.air`, `.vb`, `.wsf`	Utiliza el formato PE (Portable Executable). Las **DLL** son bibliotecas de enlace dinámico que contienen funciones compartidas por varios programas.
Linux	ELF (sin extensión específica)	Los archivos ejecutables se identifican por el **permiso de ejecución**, no por la extensión. También existen scripts con `#!` que indican el intérprete.
macOS	`.app` (contenedor Mach-O)	El `.app` es en realidad una carpeta con estructura interna que contiene el binario y los recursos.
Android	`.apk`	Archivo comprimido (ZIP) que incluye código, librerías, manifiesto y recursos.
iOS	`.ipa`	Paquete similar al APK, pero firmado digitalmente para instalarse solo mediante App Store o iTunes.

> 💬 **Comentario del profesor:**
> En **Android**, ya no se pueden subir archivos APK directamente a la Play Store; ahora se utilizan los **AAB (Android App Bundles)**.
> En Windows, los **DLL** son ejecutables especiales que se cargan por otros programas.
> En Linux, los ejecutables no necesitan extensión, solo permisos adecuados.

7️⃣ Funcionamiento de un APK

Estructura interna:

- `classes.dex` : contiene el código compilado (bytecode Dalvik/ART).
- `AndroidManifest.xml` : define permisos, actividades, servicios y metadatos.
- `res/` : carpeta con recursos (imágenes, textos, layouts).
- `lib/` : librerías nativas (C/C++).
- `META-INF/` : contiene las firmas digitales del desarrollador.

Proceso de ejecución:

1. El usuario instala el APK o lo lanza directamente.
2. El sistema verifica la **firma digital**.
3. Se leen los permisos definidos en el manifiesto.
4. Android carga el código DEX y las librerías requeridas.
5. Si faltan permisos o el usuario los deniega, ciertas funciones no podrán ejecutarse.

Ejemplo:
Una app de cámara que requiere permisos de almacenamiento y cámara fallará o limitará funciones si el usuario los rechaza.

> 💬 **Comentario del profesor:**
> Un **APK** es un **paquete que contiene todo lo necesario para lanzar una aplicación**: el código ya compilado, librerías, permisos y recursos.
> Si el usuario **deniega permisos**, la app puede funcionar parcialmente o no hacerlo en absoluto.

8️⃣ Ejecución y Tipos de Archivos Ejecutables

🔹 Ejecución de archivos según el sistema operativo

Windows:
La ejecución es directa mediante doble clic. El sistema lanza el archivo ejecutable (.exe o similar) y automáticamente inicia el proceso en segundo plano que gestiona los recursos necesarios (RAM, CPU, librerías dinámicas, etc.).
El sistema operativo interpreta el encabezado PE (Portable Executable) y crea un proceso en memoria, cargando todas sus dependencias antes de ejecutar la primera instrucción del programa.

Linux:
En los sistemas tipo Unix, la ejecución requiere permisos explícitos.

- El usuario debe otorgar permiso de ejecución mediante `chmod +x archivo` .
- Luego se ejecuta desde la consola con `./archivo` .
- Algunos ejecutables requieren privilegios administrativos (superusuario), en cuyo caso se antepone `sudo` .
  Los ejecutables en Linux suelen estar en formato **ELF (Executable and Linkable Format)**, más modular que el formato PE de Windows, ya que permite definir secciones de código, datos y librerías compartidas.

macOS:
Está basado en Unix (BSD), y utiliza el formato **Mach-O (Mach Object)**, muy similar al ELF.

Los ejecutables se distribuyen como paquetes `.app`, que en realidad son **directorios contenedores** que incluyen el binario y sus recursos. macOS, al igual que Linux, necesita permisos de ejecución para lanzarlos.

> 💬 **Comentario del profesor:**
> Cada sistema operativo gestiona los ejecutables de forma diferente.
>
> - **Windows:** inicia procesos automáticamente en segundo plano al ejecutar un `.exe`.
> - **Linux:** requiere que el usuario ejecute comandos manuales en la terminal, normalmente con privilegios de administrador.
> - **macOS:** comparte gran parte de la estructura de Linux, ya que está basado en Unix.
>   Por tanto, la representación y gestión de los ejecutables depende del **modelo interno del sistema operativo**.

🔹 Tipos de archivos ejecutables

Los archivos ejecutables se clasifican según su **portabilidad**, **dependencia del sistema operativo** y **grado de automatización**.

1. Archivos ejecutables portables

Son aquellos que no necesitan instalación y pueden ejecutarse directamente desde cualquier ubicación o dispositivo externo (por ejemplo, un pendrive).

Contienen todos los recursos necesarios dentro de la misma carpeta y no dependen del registro ni de rutas del sistema.

2. Archivos ejecutables no portables

Son dependientes del sistema operativo o versión específica.

Están diseñados para ejecutarse en una plataforma concreta y utilizan librerías o componentes del propio sistema (por ejemplo, `.exe` y `.dll` en Windows o binarios ELF específicos en distribuciones Linux).

Por esa razón, existen **más versiones disponibles** de este tipo de ejecutables, ya que cada versión del sistema puede requerir una compilación diferente o librerías concretas.

Esto explica por qué un programa de Windows 10 puede no funcionar en Windows 7 o en Linux, salvo que se utilicen capas de compatibilidad como *Wine*.

3. Archivos autoejecutables

Son ejecutables diseñados para iniciar una rutina automáticamente sin intervención del usuario.
Suelen ser instaladores o scripts que lanzan procesos en cadena (por ejemplo, un instalador que al ejecutarse descomprime y configura archivos sin pasos adicionales).

En Windows, suelen tener extensión `.exe` o `.bat`. En Linux, pueden ser scripts `.sh` con permisos de ejecución.

> 💬 **Comentario del profesor:**
> Los **ejecutables no portables** son los más comunes, y existen múltiples versiones de ellos **según el sistema operativo y la arquitectura (32 o 64 bits)**.
> Linux tiene su propia organización interna, con **directorios dedicados** a almacenar ejecutables como `/bin`, `/usr/bin`, `/sbin`, o `/usr/local/bin`.
> No cualquier archivo puede ser ejecutable: debe tener permisos adecuados y estar en formato binario o

> script válido para su entorno.
> En cambio, en Windows, cualquier archivo con una extensión reconocida por el sistema puede intentar ejecutarse, aunque no necesariamente lo logre si falta una dependencia.

🐧 2.3 Tipos de Archivos Ejecutables en Linux

En Linux no existe una extensión específica que defina a un archivo ejecutable.
Lo que determina si un archivo puede ejecutarse es **su atributo de permiso de ejecución** y su **contenido binario o interpretable**.

Ubicación estándar de los ejecutables en Linux:

- `/bin`: binarios esenciales del sistema (accesibles a todos los usuarios).
- `/usr/bin`: programas de usuario y comandos no esenciales.
- `/sbin` y `/usr/sbin`: utilidades del sistema y administración, accesibles normalmente al usuario root.
- `/usr/local/bin`: binarios instalados manualmente por el usuario o compilados desde código fuente.

Usuarios relacionados con los ejecutables:

- `root`: administrador del sistema, con permisos totales sobre todos los archivos.
- `daemon`: ejecuta servicios del sistema en segundo plano.
- `bin` y `sys`: gestionan procesos del sistema operativo.

Ejecución con permisos:
Para ejecutar un archivo desde consola:

```
chmod +x programa
./programa
```

Si el ejecutable requiere privilegios administrativos:

```
sudo ./programa
```

> 💬 **Comentario del profesor:**
> En **Linux**, no cualquier fichero puede ser un ejecutable.
> Solo aquellos que tienen permisos de ejecución y una estructura reconocida por el sistema.
> Los directorios como `/bin` o `/usr/bin` contienen los binarios del sistema, mientras que los usuarios pueden instalar sus propios ejecutables en `/usr/local/bin`.
> La **seguridad del sistema** depende de estos permisos, ya que evitan que programas no autorizados se ejecuten sin control.

🧩 Conclusión

- La **forma de ejecutar archivos** varía entre sistemas operativos:
  - Windows automatiza el proceso mediante doble clic y lanza procesos en segundo plano.
  - Linux requiere comandos manuales y permisos de ejecución.
  - macOS usa paquetes `.app` basados en Unix.
- Los **tipos de ejecutables** se diferencian por su portabilidad y dependencia del sistema operativo.
  - Los **no portables** son los más comunes y existen en múltiples versiones adaptadas a cada SO.

- En **Linux**, los ejecutables se ubican en directorios específicos (`/bin` , `/usr/bin` , `/usr/local/bin`) y su ejecución depende de los permisos de usuario.
- **No todos los archivos pueden ser ejecutables**: deben tener permisos adecuados, un formato válido y pertenecer a un entorno de ejecución reconocido.
- El conocimiento de cómo cada SO gestiona sus ejecutables es esencial para comprender la **planificación de procesos** y la **seguridad del sistema**, temas que se abordarán en la siguiente unidad.
